Exploring adaptive program behavior

Bonnichsen, Lars Frydendal; Probst, Christian W.

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

Exploring adaptive program behavior

Lars Frydendal Bonnichsen and Christian W. Probst
Technical University of Denmark

Project numer #295440

Motivation

- Modern computer systems are ever more complex.
  - Their bottlenecks can change at run time.
  - The dynamic nature makes it more difficult to make good optimization decisions.
- Adaptive behavior has been suggested as a solution.
  - Decisions are made at runtime by switching between program **versions**.

Version: implementation of part of a program.

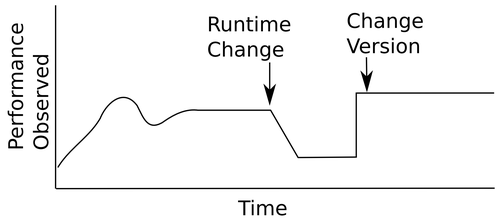

Figure : Adapting behavior based on observed performance.

- Adaptive behavior grants the optimization process more information, but comes at a cost.

Gained information	Costs
Run time bottlenecks	Monitoring and indirections
Input data	Switching versions
Problem size	Finding versions

- We need to explore the design space of adaptive behavior, to strike a good balance.

Approach

- There are different costs and gains for different forms of adaptive behavior:
  - For instance, recompilation, dispatchers, and iterative compilation.
- We want to integrate and evaluate different forms of adaptive behavior.
- Our strategy has 3 steps:
  1. Design an extensible adaptivity framwork.
  2. Expose adaptivity uniformly.
  3. Use the extensible nature of the framework to support a wide range of adaptive behavior.

Adaptivity framework

- We will optimize based on feedback, both at run time and compile time.
- We will use a uniform design to explore adaptive program behavior, as outlined in below:

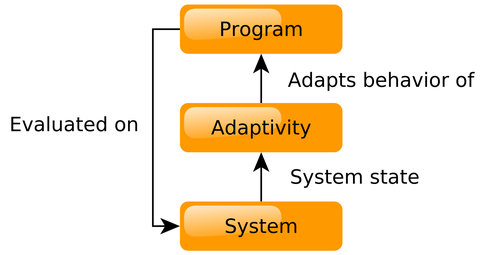

Figure : Adaptivity as a feedback loop

- Programs are iteratively refined based on feedback from the system.

Adaptivity operation

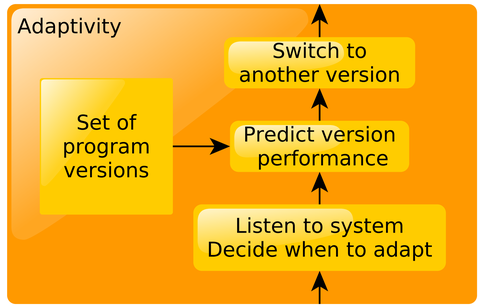

Figure : Detailed view of adaptivity

- The adaptivity block decides when to adapt, selects a version to use, and switches to it.
- The actions form a wide design space.

Design space exploration

- We charaterize the adaptivities design space by 4 factors.
  Adaptivity Action: how to adapt. **Adaptivity Trigger:** causes for adapting.
  Performance Prediction: predict versions performance. **Version Sources:** sources of alternatives.

Figure : Adaptivity design space

- The design space can encode many existing forms of adaptivity, such as:
  - Iterative compilation: (Process Forking, Once, Measure, Compiler Derived)
  - Auto-tuning: (Control flow, Once, Measure, Programmer Provided and Parameterized)
- We have a proof-of-concept adaptivity framework that would be encoded as:
  (Function Dispatch, Scheduled, Measure and Heuristic, Programmer Provided)
- We will continouesly expand the framework to explore the design space.

Contributions

- Designed an extensible adaptivity framework.
- Implemented a function level adaptivity framework.
- Outlined how the framework can be expanded to the whole design space.
- Defined criterias to evaluate adaptive program behavior.

Future work

- Using the outlined structure, we will integrate different forms of adaptive behavior to determine.
  1. How much can be gained from the adaptive behavior. ($\sim 5 - 16\%$ [1])
  2. The cost of switching versions.
  3. The overhead of supporting switches. (Best case $\sim 2\%$ [3])
  4. How should versions be selected.

At a high level, we want to determine when the benefits of adaptive behavior outweight the disadvantages.

Related Work

[1] Voss *et al*. High-Level Adaptive Program Optimization with ADAPT. In *PPOPP'01*.
[2] Fursin *et al*. A Practical Method For Quickly Evaluating Program Optimizations. In *HiPEAC'05*.
[3] Mars *et al*. Scenario Based Optimization: A Framework for Statically Enabling Online Optimizations. In *CGO'09*.

Conclusion

- Adaptive behavior allows for further optimization, but it comes at a cost.
- We need to determine when adaptive program behavior is worth the cost.
- We have a plan how to integrate different forms of adaptivity.